Tallinna Tehnikaülikool
Informaatikainstituut

Teek "vararray"

Keel C ja objektorienteeritud programmeerimine - projekt
IDK3442

Juhendaja Kaarel Allik

Koostaja Erki Suurjaak
Õpperühm LAP52
Matrikkel 970772

Tallinn 2000

Sisukord

Sissejuhatus 3

Teegi info....3

Üldinfo....3

Konstandid....4

Konstrueerimine ja destrueerimine....4

Meetodid....6

Atribuudid....7

Muu info 8

Programmi lähtetekst....9

class TVariable 10

class TVariableFiled : public TVariable....14

class TVariableArray....15

int main (void) 19

Sissejuhatus

Antud projekt on õppeaine "Keel C ja objektorienteeritud programmeerimine - projekt" (IDK3441) tarbeks koostatud projekt.

Programm on kirjutatud programmeerimiskeskkonnas Borland Turbo C++ 1.00.

Programm koosneb teegist "vararray" ja teegi mõne võimaluse näitamiseks kirjutatud funktsioonist `int main ()`.

Teegi info

Üldinfo

Teegi "vararray" keskseks klassiks on TVariableArray. Seda klassi saab kasutada muutujaväärtuste massiivi haldamiseks, faili kirjutamiseks ja failist lugemiseks. Eeskätt on ta mõeldud konfiguratsioonimuutujate säilitamiseks ja konfiguratsioonifaili haldamiseks.

Klass hoiab muutujaid kujul `char *`. Kui muutuja on sisult arv, siis on teegi kasutaja mureks vajalikud tüübiteisendused teha.

Klassi TVariableArray kuuluva objekti konstrueerimisel ettetulevate vigade korral (näiteks : spetsifitseeritav massiivi suurus ei sobi; muutujate piiranguparameetrid ei sobi omavahel) väljutakse programmist veakoodiga 1.

Klass annab võimaluse muutujaväärtuste kontrollimiseks. Iga muutuja kohta käib neli piiranguparameetrit - Mask, Length, FixedLength ja Type.

- Mask (kujul `char *`) määrab ära muutujas esineda võivad sümbolid. Näiteks Mask `"0123456789"` tähendab, et muutuja võib sisaldada ainult numbreid. Mask `""` tähendab, et muutujas võivad esineda kõik sümbolid.
- Length (kujul `int`) määrab ära muutuja maksimaalse pikkuse. Length `0` tähendab, et muutuja pikkus pole piiratud.
- FixedLength (kujul `bool`) määrab ära muutuja pikkuse fikseerituse. FixedLength `true` tähendab, et muutuja pikkus peab olema kas 0 või Length. FixedLength `false` tähendab, et muutuja pikkus ei ole fikseeritud. NB! Korraga ei tohi olla Length `0` ja FixedLength `true`.
- Type (kujul `int`) määrab ära muutuja sisulise tüübi. Näiteks Type `gvaTypeDate` tähendab, et muutuja on kuupäev kujul AAAAKKPP. Sellisel

juhul peab muutuja Mask olema kujul `"0123456789"`, Length `8` ja FixedLength `true`. (Kui muutuja piiranguparameetrid omavahel kokku ei sobi, siis väljutakse programmist käitusveaga.) Muutujale tehakse ka sisuline kontroll - kontrollitakse, kas muutuja väärtuseks antav string on reaalne kuupäev. Type `gvaTypeOther` ei sea muutuja sisule piiranguid.

Konstandid

const gvaTypeOther	- Sisulise piiranguta tüüp
const gvaTypeNone	- Sisulise piiranguta tüüp
const gvaTypeDate	- Sisult kuupäev (kujul AAAAKKPP)
const gvaTypeTime	- Sisult kellaaeg (kujul TTMM)
const gvaTypeLongTime	- Sisult sekunditega kellaaeg (kujul TTMMSS)
const gvaLengthNone	- Piiramata pikkusega
const gvaLengthOther	- Piiramata pikkusega
const gvaLengthDate	- Kuupäeva pikkusega (8)
const gvaLengthTime	- Kellaaja pikkusega (4)
const gvaLengthLongTime	- Sekunditega kellaaja pikkusega (6)
const char * const gvaMaskNumber	- Lubatud on ainult numbrid
const char * const gvaMaskEmpty	- Kõik on lubatud
const char * const gvaMaskNone	- Kõik on lubatud

Konstrueerimine ja destrueerimine

Klassi TVariableArray konstruktorit on üledefineeritud, andes võimaluse konstrueerida mõneti erinevaid muutujamassiive (mida näiteks ei kirjutata faili). Konstrueerimisel läheb vaja nelja lisatüüpi.

Eelinfo :
HandleInFile on muutuja tähis failis (näiteks "HeadingBackgroundColor"). Faili kirjutamisel kirjutatakse kõigepealt muutuja tähis, siis '=' ja siis muutuja väärtus. HandleInFile peab olema vähemalt ühe sümboli pikkune. DefaultValue on muutuja algväärtus. Kui konstrueerimisel leitakse, et algväärtus ei sobi piirangutega, siis väljutakse programmist käitusveaga.

Kõige tavalisem tüüp - kirjutatakse faili, on algväärtus.

```
class TInitVariableNormal
{
  public :

  const char * Mask;
  int Length;
  bool FixedLength;
  int Type;
  const char * DefaultValue;
```

```
  const char * HandleInFile;
};
```

Faili ei kirjutata, on algväärtus.

```
class TInitVariableNoFile
{
  public :

  const char * Mask;
  int Length;
  bool FixedLength;
  int Type;
  const char * DefaultValue;
};
```

Faili ei kirjutata, pole algväärtust.

```
class TInitVariableNoFileNoDefault
{
  public :

  const char * Mask;
  int Length;
  bool FixedLength;
  int Type;
};
```

Kirjutatakse faili, pole algväärtust.

```
class TInitVariableNoDefault
{
  public :

  const char * Mask;
  int Length;
  bool FixedLength;
  int Type;
  const char * HandleInFile;
};
```

Iga struktuuri jaoks on erinev konstruktor. ArraySize on massiivi Variables suurus (1..32767 elementi). Et konstruktoril pole kuidagi võimalik kontrollida, kas massiiv Variables on tõesti ArraySize elementi suur, siis on soovitav valida õige suurus.

Kui failinimi on antud, siis loetakse failist muutujate väärtused sisse.

```
TVariableArray (const TInitVariableNormal * const Variables, const ArraySize,
                const char * const FileName);
TVariableArray (const TInitVariableNoFile * const Variables, const ArraySize);
TVariableArray (const TInitVariableNoFileNoDefault * const Variables, const
                ArraySize);
TVariableArray (const TInitVariableNoDefault * const Variables, const ArraySize,
                const char * const   FileName);
```

Klassid TInitVariableNormal, TInitVariableNoFile, TInitVariableNoFileNoDefault, TInitVariableNoDefault on sarnased - kõigis neis on olemas liikmed Mask, Length, FixedLength ja Type. Sisuliselt võiks seetõttu teha teised

klassid TInitVariableNoFileNoDefault järglasteks. Sellega kaoks ära ka vajadus TVariableArray konstruktori üledefineerimise järele - konstruktori saaks kirja panna kujul

```
TVariableArray (const TInitVariableNoFileNoDefault * const Variables, const
                ArraySize, const char * const FileName = NULL);
```

Ja siis seda, kas tegu on tõesti klassi TInitVariableNoFileNoDefault objektide massiiviga või temast tuletatud klassi objektide massiiviga, seda saaks antud C++ kompilaatoris teha kindlaks nii, et luuakse abstraktne klass, mille järglane TInitVariableNoFileNoDefault oleks :

```
enum TypesOfVariables {InitVariableNormal, InitVariableNoFile,
InitVariableNoFileNoDefault, InitVariableNoDefault};

class TInitVariableBase
{
public:
  virtual enum TypesOfVariables GetType() = 0;
};
```

Ja klassis TInitVariableNoFileNoDefault defineeritaks `GetType ()` kui funktsioon, mis tagastab väärtuse InitVariableNoFileNoDefault, ja tema järglastes samuti kui funktsioon, mis tagastab klassile vastava tüübi.

Antud juhul aga nii pole toimitud - nimelt ei saaks sellisel juhul teegi kasutaja muutujatemassiivi konstrueerimiseks vajaminevat struktuuridemassiivi siis enam lihtsalt algväärtustada, näiteks nii :

```
TInitVariableNoDefault InitArray[Elements] =
    {
       {gvaMaskNumber, gvaLengthDate, true, gvaTypeDate, "Date"},
       {gvaMaskNumber, gvaLengthTime, true, gvaTypeTime, "Time"},
       {gvaMaskNumber, gvaLengthLongTime, true, gvaTypeLongTime, "BTime"},
       {gvaMaskNone, gvaLengthNone, false, gvaTypeOther, "String"},
    };
```

vaid ta peaks massiivi iga elemendi igat liiget eraldi väärtustama. Võib loomulikult antud klassidele teha konstruktorid, aga siis ta peaks ikkagi massiivi igat elementi eraldi väärtustama. See aga ei ole soovitav.

Klassi TVariableArray destruktoris kirjutatakse muutujate väärtused faili (kui fail on kasutuses, muidugi) ja vabastatakse hõivatud mälu.

Meetodid

`bool SetValue (int, char*)`

Annab muutujale numbriga int väärtuse char *, kui char * kvalifitseerub selleks muutujaks. Kui char * ei kvalifitseeru või antud numbriga muutujat pole, siis tagastab väärtuse `false` (muidu `true`).

`bool SetFileName (char *)`

Paneb muutujatemassiivile uue failinime ja tagastab väärtuse `true`. Kui operatsioon mingil põhjusel ebaõnnestus (näiteks massiivi ei kirjutatagi faili), siis tagastab väärtuse `false`.

`bool ReadFromFile (char *)`

Loeb muutujad etteantud nimega failist sisse. Kui operatsioon mingil põhjusel ebaõnnestus (näiteks antud faili pole või muutujamassiivi ei peagi peagi faili kirjutama/lugema), siis tagastab väärtuse `false` (muidu `true`).

`bool ReadFromFile ()`

Loeb muutujad muutujatemassiivi failist sisse. Kui operatsioon mingil põhjusel ebaõnnestus (näiteks faili pole või muutujamassiivi ei peagi faili kirjutama/lugema), siis tagastab väärtuse `false` (muidu `true`).

`bool WriteToFile (char *) const`

Kirjutab muutujad etteantud nimega faili. Kui operatsioon mingil põhjusel ebaõnnestus (näiteks antud faili pole või muutujamassiivi ei peagi peagi faili kirjutama/lugema), siis tagastab väärtuse `false` (muidu `true`).

`bool WriteToFile () const`

Kirjutab muutujad muutujatemassiivi faili. Kui operatsioon mingil põhjusel ebaõnnestus (näiteks faili pole või muutujamassiivi ei peagi peagi faili kirjutama/lugema), siis tagastab väärtuse `false` (muidu `true`).

Atribuudid

`bool GetFixedLength (int) const`

Tagastab muutuja #int fikseerituse (kas muutuja pikkus on konstantne). Kui int-indat muutujat pole, tagastab väärtuse `false` (siin on muidugi halb see, et ei saa kuidagi kontrollida, kas `false` käib muutuja fikseerituse kohta või on veakood. Aga seda, kas muutuja #int eksisteerib, saab kontrollida muude atribuutidega (näiteks `char* GetMask (int) const` -ga - kui tagastatud väärtus on `NULL`, siis muutujat ei eksisteeri).).

`char* GetMask (int) const`

Tagastab muutuja #int sümbolmaski (milliseid sümboleid muutuja sisaldada võib). Kui antud muutujat pole, tagastab väärtuse `NULL`.

`char * GetHandle (int) const`

Tagastab muutuja #int tunnuse failis. Kui muutujal seda pole (muutujatemassiivi ei kirjutata faili), siis tagastab väärtuse NULL.

`int GetLength (int) const`

Tagastab muutuja #int maksimaalse pikkuse. Kui antud muutujat pole, siis tagastab väärtuse `NULL`. Siinkohal on jällegi see halb asi, et kui muutujal maksimaalset pikkust pole, siis tagastatakse samuti `NULL`.

`char * operator[] (int) const`

Tagastab muutuja #int väärtuse. Kui antud muutujat pole, siis tagastab väärtuse `NULL`.

`char * GetValue (int) const`

Tagastab muutuja #int väärtuse. Kui antud muutujat pole, siis tagastab väärtuse `NULL`. Täiesti ekvivalentne `char * operator[] (int)` - ga.

Muu info

Teegis "vararray" on veel defineeritud klassid TVariable ja selle järglane TVariableFiled. Klass TVariableArray kasutab neid klasse muutujatemassiivi hoidmiseks. Siinkohal pole nende klasside täpsemat spetsifikatsiooni ära toodud - neid on vaja ainult klassil TVariableArray. Lühidalt, klassi TVariable objektis säilitatakse kogu info ühe muutuja kohta, samuti teostatakse seal kontroll muutuja väärtuse sobilikkusest muutujale seatud piirangutega. Klassis TVariableFiled on veel juures lisaliikmed faili jaoks (muutuja tähis failis ja muu sellega seonduv). Täpsemat infot on võimalik saada programmi tekstis sisalduvatest kommentaaridest.

Teegis kasutatakse standardteekide kõrval veel teeki `"utils.cpp"`. Antud teegis on tüübi `bool` definitsioon ning järgmised funktsioonid :

`bool IsDate (char *)`

Kontrollib etteantud stringi sobivust kuupäevaks kujul AAAAKKPP.

`bool IsTime (char *)`

Kontrollib etteantud stringi sobivust kellaajaks kujul TTMM.

`bool IsLongTime (char *)`

Kontrollib etteantud stringi sobivust kellaajaks kujul TTMMSS.

`void FatalError (const char *)`

Kirjutab etteantud stringi voogu `cerr` ja väljub programmist veakoodiga 1 (`exit(1)`).

NB! Kui kasutatakse mingit funktsiooni, mis tagastab väärtuse char *, siis tuleb see ka ise kustutada.

Programmi lähtetekst

```
#include <string.h>
#include <fstream.h>
#include <stdlib.h>
#include "utils.cpp"


// ------------------------------ Konstandid -----------------------------


const gvaTypeOther    =  0;   // Sisulise piiranguta tüüp
const gvaTypeNone     =  0;   // Sama asi, teine nimi
const gvaTypeDate     =  1;   // Kuupäev (aaaakkpp)
const gvaTypeTime     =  2;   // Kellaaeg (ttmm)
const gvaTypeLongTime =  3;   // Kellaaeg sekunditega (ttmmss)

const gvaLengthNone     = 0;  // Piiranguteta pikkus
const gvaLengthOther    = 0;  // Sama asi, teine nimi
const gvaLengthDate     = 8;  // Kuupäeva pikkus
const gvaLengthTime     = 4;  // Kellaaja pikkus
const gvaLengthLongTime = 6;  // Sekunditega kellaaja pikkus

const char * const gvaMaskNumber = "0123456789";     // Numbrite mask
const char * const gvaMaskEmpty  = "";               // Piiranguteta mask
const char * const gvaMaskNone   = "";               // Sama asi, teine nimi


// ------------------------------ /Konstandid -----------------------------



// ---------------- Vajalikud struktuurid initsialiseerimiseks -------------



// Kõige tavalisem tüüp - kirjutatakse faili, on piirangud, on vaikeväärtus
class TInitVariableNormal
{
  public :

  const char * Mask;         // Muutuja lubatud sümbolid stringina
  int Length;                // Muutuja maksimaalne pikkus
  bool FixedLength;          // Muutuja pikkuse fikseeritus
  int Type;                  // Muutuja tüüp; gvaTypeDate, gvaTypeOther vms
                             // Muud reeglid peavad siis ka vastama tüübile
  const char * DefaultValue; // Muutuja algväärtus
  const char * HandleInFile; // Muutuja tähis failis
};


// Faili ei kirjutata, on piirangud, on vaikeväärtus
class TInitVariableNoFile
{
  public :

  const char * Mask;         // Muutuja lubatud sümbolid stringina
  int Length;                // Muutuja maksimaalne pikkus
  bool FixedLength;          // Muutuja pikkuse fikseeritus
  int Type;                  // Muutuja tüüp; gvaTypeDate, gvaTypeOther vms
                             // Muud reeglid peavad siis ka vastama tüübile
  const char * DefaultValue; // Muutuja algväärtus
};
```

```
// Faili ei kirjutata, on piirangud, pole vaikeväärtust
class TInitVariableNoFileNoDefault
{
  public :

  const char * Mask;         // Muutuja lubatud sümbolid stringina
  int Length;                // Muutuja maksimaalne pikkus
  bool FixedLength;          // Muutuja pikkuse fikseeritus
  int Type;                  // Muutuja tüüp; gvaTypeDate, gvaTypeOther vms
};                           // Muud reeglid peavad siis ka vastama tüübile


// Kirjutatakse faili, on piirangud, pole vaikeväärtust
class TInitVariableNoDefault
{
  public :

  const char * Mask;         // Muutuja lubatud sümbolid stringina
  int Length;                // Muutuja maksimaalne pikkus
  bool FixedLength;          // Muutuja pikkuse fikseeritus
  int Type;                  // Muutuja tüüp; gvaTypeDate, gvaTypeOther vms
                             // Muud reeglid peavad siis ka vastama tüübile
  const char * HandleInFile; // Muutuja tähis failis
};



// --------------- /Vajalikud struktuurid initsialiseerimiseks -------------



// ------------------------------- TVariable -----------------------------



class TVariable
{
  protected :


  char* Mask_;         // Muutuja lubatud sümbolid stringina
  int Length_;         // Muutuja maksimaalne pikkus
  bool FixedLength_;   // Muutuja pikkuse fikseeritus
  int Type_;           // Muutuja tüüp; gvaTypeDate, gvaTypeOther vms
                       // Muud reeglid peavad siis ka vastama tüübile
  char* Value_;        // Muutuja väärtus

                       // Konstrueerimisel kasutatav funktsioon, mis
                       // kontrollib parameetrite kokkusobivust
  inline void ConstructionCheck
                        (const char * Mask, const int Length,
                         const bool FixedLength, const int Type,
                         const char * DefaultValue);


  public :

  TVariable (const char * Mask, const int Length, const bool FixedLength,
             const int Type, const char * DefaultValue = NULL);

  virtual ~TVariable ();

  // Need kaks funktsiooni on defineeritud selleks, et kui TVariableArray-s
  // on TVariable-massiiv loodud tegelikult TVariableFiled-idest, et siis
  // ei peaks upcastingut tegema.
  virtual char * GetHandle () const {return NULL;};
  virtual char * ForWritingToFile () const {return NULL;};
```

```
  const bool SetValue (const char * NewValue);   // Uue väärtuse andmine
  char * GetValue () const;                      // Väärtuse võtmine
  char * GetMask () const;                       // Muutuja maski võtmine
  const GetLength () const;                      // Muutuja maks.pikkuse võtmine
  const bool GetFixedLength () const;            // Muutuja pikkuse fikseerituse võtmine
  const GetType () const;                        // Muutuja tüübi saamine

  const bool IsValid (const char * Value) const; // Value sobilikkuse kontroll
};



void TVariable::ConstructionCheck
          (const char * Mask, const int Length,
           const bool FixedLength, const int Type,
           const char * DefaultValue)
{
     int ErrorFound = 0;
     switch (Type)             // Kontroll tuntud tüübi järgi
        {
        case gvaTypeDate :     if (Mask != gvaMaskNumber     // Kui maksimum-
                                 || Length != gvaLengthDate) // pikkus on
                                   ErrorFound = 1;           // erinev tuntust
                               if (strlen(DefaultValue) != 0)
                                 if (!IsDate(DefaultValue))  // Kui vaikeväärtus
                                   ErrorFound = 2;           // pole see tüüp
                               if (FixedLength != true)      // Pikkus PEAB
                                   ErrorFound = 1;           // olema fiks.
                               break;
        case gvaTypeTime :     if (Mask != gvaMaskNumber
                                 || Length != gvaLengthTime)
                                   ErrorFound = 1;
                               if (strlen(DefaultValue) != 0)
                                 if (!IsTime(DefaultValue))
                                   ErrorFound = 2;
                               if (FixedLength != true)      // Pikkus PEAB
                                   ErrorFound = 1;           // olema fiks.
                               break;
        case gvaTypeLongTime : if (Mask != gvaMaskNumber
                                 || Length != gvaLengthLongTime)
                                   ErrorFound = 1;
                               if (strlen(DefaultValue) != 0)
                                 if (!IsLongTime(DefaultValue))
                                   ErrorFound = 2;
                               if (FixedLength != true)      // Pikkus PEAB
                                   ErrorFound = 1;           // olema fiks.
                               break;
        default : break;
        }
     if (ErrorFound == 1) FatalError ("\nTVariable : Length or mask restriction conflicts
with data type.\n");
     if (ErrorFound == 2) FatalError ("\nTVariable : Default value conflicts with data
type.\n");

                                // Kui pikkus on konstantne, siis maksimum
     if (!Length && FixedLength)  // peab fikseeritud olema
            FatalError ("\nTVariable : If length is fixed, maximum length must be fixed
also.\n");


     if (Type == gvaTypeOther && strlen(DefaultValue) != 0)
      {
      if (Length)           // Kui maksimaalne pikkus on fikseeritud
       if (FixedLength)     // Kui pikkus on konstantne
         {
         if ((strlen(DefaultValue) != Length))
            FatalError ("\nTVariable : Default value length conflicts with length
restriction.\n");
         }
        else                // Kui pikkus pole konstantne, siis tuleb
```

```
         {                  // vaadata, ega asi pole pikem kui lubatud
         if (strlen(DefaultValue) > Length)
           FatalError ("\nTVariable : Default value length is longer than allowed
maximum.\n");
         };

      if (strlen(Mask))     // Kui Mask on tühi, siis piiranguid pole
        {
        for (int p = 0; p < strlen (DefaultValue); p++)   // Vastasel juhul
          if (!strchr (Mask, DefaultValue[p]))            // jupiti kontroll
            FatalError ("\nTVariable : Default value conflicts with mask restriction.\n");
        }
      }

     return;
}


TVariable::TVariable (const char * Mask, const int Length,
                      const bool FixedLength, const int Type,
                      const char * DefaultValue)
{
     ConstructionCheck (Mask, Length, FixedLength, Type, DefaultValue);

     Mask_ = new char[strlen(Mask) + 1];
     strcpy (Mask_, Mask);

     Value_ = new char[strlen(DefaultValue) + 1];
     strcpy (Value_, DefaultValue);

     Length_ = Length;

     FixedLength_ = FixedLength;

     Type_ = Type;

     return;
}


TVariable::~TVariable ()
{
    delete (Mask_);
    delete (Value_);
    return;
}


const bool TVariable::SetValue (const char * NewValue)
{
    if (!IsValid(NewValue))            // Kui uus väärtus on konfliktis
      return false;                    // piirangutega, siis jääb asi katki.
    if (strcmp(NewValue, Value_))      // Kui uus on sama, mis eelmine, siis
       {                               // pole mõtet vaeva näha
       delete (Value_);
       Value_ = new char[strlen(NewValue) + 1];
       strcpy (Value_, NewValue);
       }
    return true;
}


char * TVariable::GetValue () const
{
   char * tmpValue = new char[strlen(Value_) + 1];
   strcpy (tmpValue, Value_);
   return tmpValue;
}


char * TVariable::GetMask () const
```

```
{
   char * tmpValue = new char[strlen(Mask_) + 1];
   strcpy (tmpValue, Mask_);
   return tmpValue;
}


const TVariable::GetLength () const
{
   return Length_;
}


const bool TVariable::GetFixedLength () const
{
   return FixedLength_;
}


const TVariable::GetType () const
{
   return Type_;
}


const bool TVariable::IsValid (const char * Value) const
{
   switch (Type_)
      {
      case gvaTypeDate :     if (strlen(Value) != 0) // Value > ""
                               if (!IsDate(Value))  // Kui vaikeväärtus
                                  return false;      // pole see tüüp
                             break;
      case gvaTypeTime :     if (strlen(Value) != 0)
                               if (!IsTime(Value))
                                  return false;
                             break;
      case gvaTypeLongTime : if (strlen(Value) != 0)
                               if (!IsLongTime(Value))
                                  return false;
                             break;
      case gvaTypeOther :    if (strlen(Value) != 0)
                              {
                              if (Length_) // Kui maksimaalne pikkus on olemas
                               if (FixedLength_)  // Kui pikkus on konstantne
                                {
                                if ((strlen(Value) != Length_)) return false;
                                }
                               else if (strlen(Value) > Length_)
                                   return false;
                              if (strlen(Mask_))
                               {
                               for (int p = 0; p < strlen (Value); p++)
                               if (!strchr (Mask_, Value[p]))
                                 return false;
                               }
                              }
                             break;
      default : break;
      }

   return true;
}


// ------------------------------ /TVariable -----------------------------
```

```
// ---------------------------- TVariableFiled -----------------------------



class TVariableFiled : public TVariable
{
  protected :

  char* HandleInFile_; // Muutuja tähis failis


  public :

  TVariableFiled (const char * Mask, const int Length, const bool FixedLength,
                  const int Type, const char * HandleInFile,
                  const char * DefaultValue = NULL);

  virtual ~TVariableFiled ();

  inline virtual char * GetHandle () const;        // Muutuja tähis failis
  inline virtual char * ForWritingToFile () const; // Muutuja kuju failis
};                                                 // (Tähis=Väärtus)



TVariableFiled::TVariableFiled (const char * Mask, const int Length,
                                const bool FixedLength, const int Type,
                                const char * HandleInFile,
                                const char * DefaultValue)
 : TVariable (Mask, Length, FixedLength, Type, DefaultValue)
{

     if (!strlen(HandleInFile)) FatalError ("\nTVariable : Variable handle must be at
least one char long.\n");


     HandleInFile_ = new char[strlen(HandleInFile) + 1];
     strcpy (HandleInFile_, HandleInFile);

     return;
}


TVariableFiled::~TVariableFiled ()
{
    delete (HandleInFile_);
    return;
}


inline char * TVariableFiled::GetHandle () const
{
   char * tmpValue = new char[strlen(HandleInFile_) + 1];
   strcpy (tmpValue, HandleInFile_);
   return tmpValue;
}


inline char * TVariableFiled::ForWritingToFile () const
{
 char * tmpChar = new char[strlen(HandleInFile_) + strlen(Value_) + 2];
 strcpy (tmpChar, HandleInFile_);
 strcat (tmpChar, "=");
 strcat (tmpChar, Value_);
 return tmpChar;
}
```

```
// --------------------------- /TVariableFiled -----------------------------



// --------------------------- TVariableArray -----------------------------



class TVariableArray
{
 private :

 TVariable** VarArray; // Massiiv, kus muutujaid hoitakse
 int ArraySize_;            // Massiivi suurus
 char * FileName_;          // Konfiguratsioonifaili nimi


 public :
 // Konstruktorit on overloaditud
 TVariableArray                                        // Muutujate
  (const TInitVariableNormal * const Variables, const// algväärtustamine ja
   ArraySize, const char * const FileName);          // failist lugemine
 TVariableArray                                       // Muutujate
  (const TInitVariableNoFile * const Variables,     // algväärtustamine,
   const ArraySize);                                 // failist ei loeta
 TVariableArray                                       // Muutujate
  (const TInitVariableNoFileNoDefault * const Variables,// algväärtustamine,
   const ArraySize);                                 // pole faili ega vaikeväärtust
 TVariableArray                                       // Muutujate
  (const TInitVariableNoDefault * const Variables,  // algväärtustamine,
   const ArraySize, const char * const FileName);   // pole vaikeväärtust

 ~TVariableArray ();        // Muutujate kustutamine, ehk ka failikirjutamine
 const bool SetValue (const Number,                     // Mingi elemendi
                      const char * const NewValue);     // väärtuse muutmine

 const bool SetFileName (const char * const FileName); // Failinime muutmine

 const bool GetFixedLength (const Number) const; // Mingi elemendi fikseeritus
 char* GetMask (const Number) const;        // Mingi elemendi mask
 char* GetHandle (const Number) const;       // Mingi elemendi tunnus failis
 const GetLength (const Number) const;      // Mingi elemendi maks. pikkus
 char * operator[] (const Number) const;    // Mingi elemendi väärtus
 char * GetValue (const Number) const;      // Mingi elemendi väärtus
 const bool ReadFromFile (const char        // Muutujate lugemine failist
                     * const FileName);     //  FileName
 const bool ReadFromFile ();                // Muutujate lugemine konfifailist
 const bool WriteToFile (const char         // Muutujate kirjutamine faili
                    * const FileName);      // nimega FileName
 const bool WriteToFile ();                 // Muutujate kirjutamine konfifaili
};



// Tavaline - on fail, on vaikeväärtused
TVariableArray::TVariableArray (const TInitVariableNormal * const Variables,
                                const ArraySize,
                                const char * const FileName)
{
     if (ArraySize <  1 || ArraySize > 32767)         // Kontroll ikka peal
        FatalError ("\nTVariableArray : Array can be 1..32767 elements long.\n");

     ArraySize_ = ArraySize;
     FileName_ = new char[strlen(FileName) + 1];
     strcpy (FileName_, FileName);


     VarArray = new TVariable*[ArraySize_];
```

```
     int i;
     for (i = 0; i < ArraySize_; i++)
      {
          VarArray[i] = new TVariableFiled (Variables[i].Mask,
                                            Variables[i].Length,
                                            Variables[i].FixedLength,
                                            Variables[i].Type,
                                            Variables[i].HandleInFile,
                                            Variables[i].DefaultValue);
      }

     ReadFromFile ();

     return;
}


// Pole faili, on vaikeväärtused
TVariableArray::TVariableArray (const TInitVariableNoFile * const Variables,
                                const ArraySize)
{
     if (ArraySize <= 0 || ArraySize > 32767)         // Kontroll ikka peal
        FatalError ("\nTVariableArray : Array can be 1..32767 elements long.\n");
     ArraySize_ = ArraySize;
     FileName_ = NULL;

     VarArray = new TVariable*[ArraySize_];
     int i;
     for (i = 0; i < ArraySize_; i++)
      {
          VarArray[i] = new TVariable (Variables[i].Mask,
                                       Variables[i].Length,
                                       Variables[i].FixedLength,
                                       Variables[i].Type,
                                       Variables[i].DefaultValue);
      }

     return;
}


// Pole faili, pole vaikeväärtusi
TVariableArray::TVariableArray (const TInitVariableNoFileNoDefault *
                                const Variables, const ArraySize)
{
     if (ArraySize <= 0 || ArraySize > 32767)         // Kontroll ikka peal
        FatalError ("\nTVariableArray : Array can be 1..32767 elements long.\n");
     ArraySize_ = ArraySize;
     FileName_ = NULL;

     VarArray = new TVariable*[ArraySize_];
     int i;
     for (i = 0; i < ArraySize_; i++)
      {
          VarArray[i] = new TVariable (Variables[i].Mask,
                                       Variables[i].Length,
                                       Variables[i].FixedLength,
                                       Variables[i].Type);
      }

     return;
}


// On fail, pole vaikeväärtusi
TVariableArray::TVariableArray (const TInitVariableNoDefault * const Variables,
                                const ArraySize, const char * const FileName)
{
     if (ArraySize <= 0 || ArraySize > 32767)         // Kontroll ikka peal
        FatalError ("\nTVariableArray : Array can be 1..32767 elements long.\n");
     ArraySize_ = ArraySize;
```

```
     FileName_ = new char[strlen(FileName) + 1];
     strcpy (FileName_, FileName);

     VarArray = new TVariable*[ArraySize_];
     int i;
     for (i = 0; i < ArraySize_; i++)
      {
          VarArray[i] = new TVariableFiled (Variables[i].Mask,
                                            Variables[i].Length,
                                            Variables[i].FixedLength,
                                            Variables[i].Type,
                                            Variables[i].HandleInFile);
      }

     ReadFromFile ();

     return;
}


TVariableArray::~TVariableArray (void)
{
     int i;
     WriteToFile();

     // Kõikidele muutujaobjektidele kinga
     for (i = 0; i < ArraySize_; i++)
      delete (VarArray[i]);
     delete VarArray;

     delete FileName_; // Kui ka faili pole, siis NULLi võib rõõmuga kustutada

     return;
}


const bool TVariableArray::SetValue (const Number,
                                     const char * const NewValue)
{
    if (Number < 0 || Number >= ArraySize_)     // Kui sätitav element
      return false;                             // ei kuulu massiivi

    return VarArray[Number]->SetValue (NewValue);
}


const bool TVariableArray::SetFileName (const char * const FileName)
{
    if (FileName_ == NULL) return false;  // Kui konfifaili ei peagi olema
    if (!strcmp (FileName, FileName_)) return true;

    delete FileName_;
    FileName_ = new char[strlen(FileName) + 1];
    strcpy (FileName_, FileName);

    return true;
}


const bool TVariableArray::GetFixedLength (const Number) const
{
    if (Number < 0 || Number >= ArraySize_) return false;

    return VarArray[Number]->GetFixedLength ();
}


char * TVariableArray::GetMask (const Number) const
{
    if (Number < 0 || Number >= ArraySize_) return false;
```

```
    char * tmp = VarArray[Number]->GetMask ();
    return tmp;
}


char* TVariableArray::GetHandle (const Number) const
{
    if (FileName_ == NULL) return false;  // Kui konfifaili ei peagi olema

    if (Number < 0 || Number >= ArraySize_) return false;

    char * tmp = VarArray[Number]->GetHandle();
    return tmp;
}


const TVariableArray::GetLength (const Number) const
{
    if (Number < 0 || Number >= ArraySize_) return false;

    return VarArray[Number]->GetLength ();
}


char * TVariableArray::operator[] (const Number) const
{
    if (Number < 0 || Number >= ArraySize_) return false;

    char * tmp = VarArray[Number]->GetValue ();
    return tmp;
}


char * TVariableArray::GetValue (const Number) const
{
    return operator[] (Number);
}


const bool TVariableArray::ReadFromFile (const char * const FileName)
{
     if (FileName_ == NULL) return false; // Kui faili pole, ei tohi ka lugeda

     if (strlen(FileName))  // Kui failinimi antud
      {
      ifstream ConfigFile (FileName);
      if (ConfigFile)
       {
       char Row[201];
       while (!ConfigFile.eof())
        {
        ConfigFile.getline (Row, 200);
        // getline loeb stringi sisse ka reavahe, see tuleb eemaldada
        if (Row[strlen(Row) - 1] = '\n') Row[strlen(Row) - 1] = '\x0';

        if (strchr (Row, '='))  // Kui rida on omistamine, peab seal võrdusmärk olema
         {
         for (int i = 0; i < ArraySize_; i++)
          {
          const char *HandleName = VarArray[i]->GetHandle();

          // Kui loetud stringi esimene pool ühtib mingi muutujatähisega, siis..
          if (!strncmp (Row, HandleName, strlen(HandleName)))
           {
           int tmpSize = strlen(Row) - strlen(HandleName) - 1;
           char * tmpChar = new char[tmpSize + 1];
           strncpy (tmpChar, Row + strlen(HandleName) + 1, tmpSize);
           tmpChar[tmpSize] = '\0';
           VarArray[i]->SetValue (tmpChar); // Väärtuse kontroll toimub massiivis
           delete tmpChar;
           delete HandleName;
```

```
           break;
           }
          }
         }
        }
       ConfigFile.close;
       }
      else return false;
      }
      else return false;
     return true;
};


const bool TVariableArray::ReadFromFile ()
{
     return ReadFromFile (FileName_);
};


const bool TVariableArray::WriteToFile (const char * const FileName)
{
     if (FileName_ == NULL) return false; // Kui faili pole, siis ei tohi ka
                                          // kirjutada - muutujatel pole tunnuseid
     if (strlen(FileName))
      {
      ofstream ConfigFile;
      ConfigFile.open (FileName);
      if (ConfigFile)
       {
       for (int i = 0; i < ArraySize_; i++)
        {
        char * Line = VarArray[i]->ForWritingToFile ();
        ConfigFile << Line << '\n';
        delete Line;
        }
       }
      else return false;
      ConfigFile.close;
      }
     else return false;
     return true;
};


const bool TVariableArray::WriteToFile ()
{
 return WriteToFile (FileName_);
};



// --------------------------- /TVariableArray -----------------------------



// ----------------------------- Peaprogramm -------------------------------


// Peaprogramm, klassi TVariableArray mõnede võimaluste näitamiseks.
// Programmis saab sisestada paar väärtust. Need kirjutatakse faili.
// See programm on täiesti mõttetu.
#include <conio.h>

int main (void)
{
   const Elements = 4;
   const char * const FileName = "test.cfg";
```

```
enum {DateEntered, TimeEntered, LongTimeEntered, StringEntered};

TInitVariableNoDefault InitArray[Elements] =
 {
    {gvaMaskNumber, gvaLengthDate,     true,  gvaTypeDate,     "DateEntered"},
    {gvaMaskNumber, gvaLengthTime,     true,  gvaTypeTime,     "TimeEntered"},
    {gvaMaskNumber, gvaLengthLongTime, true,  gvaTypeLongTime, "LongTimeEntered"},
    {gvaMaskNone,   gvaLengthNone,     false, gvaTypeOther,    "StringEntered"},
 };

TVariableArray Variables (InitArray, Elements, FileName);

char * TempChr = NULL;
char Key;
int StillAtIt = 1;


do
 {
 clrscr();

 TempChr = Variables[DateEntered];
 cout << "1. Enter date (YYYYMMDD). Entered date : " << TempChr << '\n';
 delete TempChr;

 TempChr = Variables[TimeEntered];
 cout << "2. Enter time (HHMM). Entered time : " << TempChr << '\n';
 delete TempChr;

 TempChr = Variables[LongTimeEntered];
 cout << "3. Enter time (HHMMSS). Entered time : " << TempChr << '\n';
 delete TempChr;

 TempChr = Variables[StringEntered];
 cout << "4. Enter string. Entered string : " << TempChr << '\n';
 delete TempChr;

 cout << "5. Exit\n\nMake your choice : ";


 Key = getch();

 switch (Key)
 {
 case '1' :
       cout << "\nEnter date : ";
       TempChr = new char[130];
       if (cin.peek() == '\n') cin.get();
       cin.get (TempChr, 128);

       if (!Variables.SetValue (DateEntered, TempChr))
        cout << "\nError! Entered value is not a date. Press any key to continue.\n",
       getch();
       delete TempChr;
       break;
 case '2' :
       cout << "\nEnter time : ";
       TempChr = new char[130];
       if (cin.peek() == '\n') cin.get();
       cin.get (TempChr, 128);

       if (!Variables.SetValue (TimeEntered, TempChr))
        cout << "\nError! Entered value is not a time. Press any key to continue.\n",
       getch();
       delete TempChr;
       break;
 case '3' :
       cout << "\nEnter long time : ";
       TempChr = new char[130];
       if (cin.peek() == '\n') cin.get();
       cin.get (TempChr, 128);
```

```
          if (!Variables.SetValue (LongTimeEntered, TempChr))
           cout << "\nError! Entered value is not a long time. Press any key to
continue.\n",
          getch();
          delete TempChr;
          break;
    case '4' :
          cout << "\nEnter string : ";
          TempChr = new char[130];
          if (cin.peek() == '\n') cin.get();
          cin.get (TempChr, 128);

          Variables.SetValue (StringEntered, TempChr);
          delete TempChr;
          break;
    case '5' :
          StillAtIt = 0;
          break;
    default : break;
    };
   } while (StillAtIt);


   clrscr();
   return 0;
}


// ---------------------------- /Peaprogramm ------------------------------
```